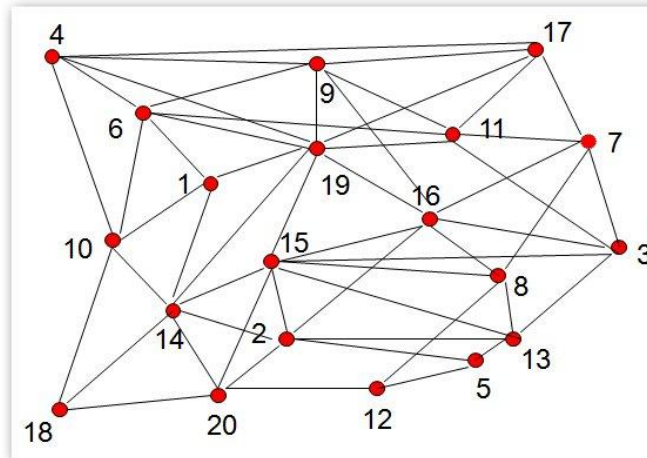


Solving the Traveling Salesman Problem

Raja Sooriamurthi

1. The Problem

The Traveling Salesman Problem is a classic problem in optimization: find the least-cost round trip route amongst N cities, visiting each city exactly once. For example, in the below figure, starting at city 1, what is the shortest route that will visit all cities and bring us back to city 1?



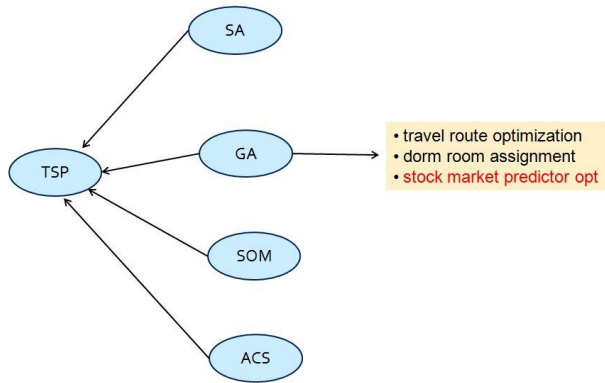
This series of assignments involves (1) solving the TSP using various machine learning techniques such as genetic algorithms, simulated annealing, SOM etc (2a) using classic prediction algorithms (knn etc) to predict the DOW and (2b) use optimization methods from the first part (e.g., GAs) to optimize the prediction of the DOW

2. Objectives

The objectives of the series of assignments are multifold:

- To use the TSP as a test bed for various ML approaches (e.g., genetic algorithms, simulated annealing, Kohonen Self Organizing Maps etc).
- To briefly study various real world applications of the TSP (e.g., logistics; genome sequencing; manufacturing: IC testing, PCB drilling; aiming telescopes; and x-ray crystallography)
- To tie in problems from machine learning to general Computer Science (e.g., revisiting the concepts of NP completeness).
- To provide a historical perspective of a problem with a rich history, significant practical applications, and numerous technological approaches.)
- To use the algorithm(s) developed for solving the TSP to solve a different optimization problem (predicting the DOW).

These objectives may be pictorially depicted as:



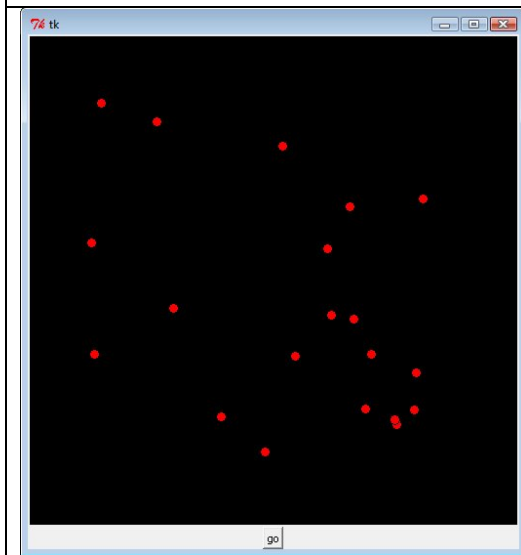
3. Description

Part 1: Solving the TSP

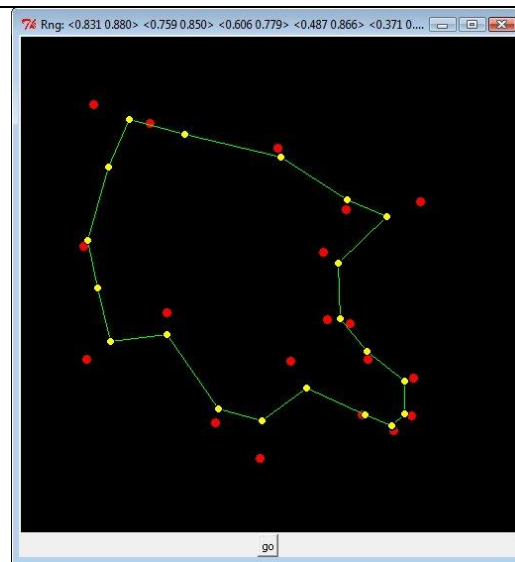
Using the descriptions of GA, SA, and SOM we've discussed in class implement these algorithms to solve instances of the TSP:

Solving the TSP Using a Genetic Algorithm	
Distribution of 20 cities with an initial random solution	Final evolved solution by the GA

Solving the TSP Using a Kohonen Self Organizing Map (SOM)



Random Distribution of 20 cities



Final solution produced by the SOM

Part 2a: Predicting the DOW Jones Industrial Average

- To gain hands-on familiarity with various prediction methods, e.g., simple averaging, statistical regression, nearest neighbor, decision trees, neural networks.

Acknowledgment: Parts 2a and 2b are modeled after a similar assignment used by Professor Zbigniew Michalewicz.

The assignment

In this assignment you will explore various algorithms to predict a stock market indicator. The goal is to devise a system that can predict with minimal least mean square (LMS) error.

The file [dowjones-20000101-20080407.csv](#) contains stock market information about the Dow Jones Industrial average taken from [Yahoo finance](#) web site. Your goal is to devise a method to predict the value of **Adj Close**.

You are free to use any of the prediction methods we have discussed in class. You are welcome to adapt code from our book as well as utilize any libraries/code you may find on the web as long as the source is properly acknowledged. The only restrictions is that all code needs to be in Python.

As a baseline everyone needs to implement a prediction method based on averages. By this method, you will predict the **adj close** for a particular day by averaging the **adj close** of the immediate previous n days. If you were to invoke this code from the command line for an averaging window size of 10 (i.e., predict by averaging the values of the the past 10 days) it would be:

```
% python avg_predict.py dowjones-20000101-20080407.csv 10
```

Other prediction methods that could be explored are k nearest neighbors and decision trees.

There are numerous parameters you can explore for each method: the averaging window-size, the size of k , how to weight each neighbor, the relative weights of each attribute etc. You will submit a short report giving details of your experiments, graphs for various parameters (e.g., n for the averaging method, k for nearest neighbor) and indicate which method with which parameters gave you the best result.

Given the range of possibilities that can be explored in this assignment you are encouraged to work in pairs (one submission per pair). I've prepared an [excel spreadsheet](#) that shows an average prediction with a window size of 10. You could use the numbers in the spreadsheet to compare against that produce by your code.

Notes

- I've got a request to provide the Dow Jones data in increasing date order. I've reversed the contents of the file [dowjones-20000101-20080407.csv](#) and have placed it at: [dowjones-20000101-20080407-reverse.csv](#)
- If you would like to optionally specify command line arguments you could use the following template for `avg_predict`:

```
import sys

# defaults
dat_file = 'dowjones-20000101-20080407.csv'
window_size = 10

# ...

if __name__ == '__main__':
    print sys.argv
    if len(sys.argv) == 1:
        # use default data file and window size
        script_name = sys.argv[0]
    elif len(sys.argv) == 2:
        script_name = sys.argv[0]
        dat_file = sys.argv[1]
    else: # len(sys.argv) >= 3
        script_name = sys.argv[0]
        dat_file = sys.argv[1]
```

```
window_size = sys.argv[2]

print script_name, dat_file, window_size
```

Part2b: An Optimal Investment Strategy

- To create an optimal investment strategy using the data-set and prediction method you identified in assignment-2

The assignment

In this assignment you will develop an optimal investment strategy based on the prediction method of the previous assignment. The general idea is the following: On each day you will receive a certain amount of money. You may choose to invest that money or keep it in a bank. At the end of the simulation period you must sell all your assets and convert everything to cash. The goal is to determine an investment strategy that will maximize your cash at the end.

The details

- You will initially start with a bank cash balance of \$0.
- Starting Jan-3-2000, you will receive \$10 per day. Each day, before the bell rings, two things will happen:
 1. \$10 will be credited to your bank account and
 2. you will make a decision as to how many shares of the Dow Jones Market indicator you will buy or sell.
- You are allowed to buy fractional quantities of the share.
- When all transactions are made in the morning, the latest available value of the market indicator will be used e.g., on the morning of Jan-27-2000, the adjusted close value on Jan-26-2000 will be used. Similarly on the morning of Jan-31-2000 the adjusted close value of Jan-28-2000 will be used.
- Any transaction (buy or sell) has a 1% transaction cost associated with it. So if you have \$10 in cash you can only buy shares for \$9.9.
- On the last day Apr-8-2008 you will sell all shares you hold based on the adjust closing price of Apr-7-2008.
- Your goal is to maximize the amount of money you would have after cashing out on Apr-8-2008.
- Your output should be produced as a CSV file with the following fields:

Date	Adj Close	Cash	Buy/Sell	Shares

The Data Sets

The following data sets are available off of the course web site:

- [dowjones-20000101-20080407.csv](#)
- [dowjones-20000101-20080407-reverse.csv](#)

What to submit

Submit all your code and a short report on what optimization strategy you used. Also submit the above CSV file and a graph based on the file that shows the changes in cash value across time and the final amount of cash you have when you cash out.

4. References

1. Programming Collective Intelligence: Building smart web 2.0 applications. Toby Segaran. O'Reilly 2007.
2. How to Solve It: Modern Heuristics. Zbigniew Michalewicz and David Fogel. Springer 2004.